Gestionnaire d'impression

Ho-Dac Phong Hanhardt Frédéric

February 14, 2002

Contents

1 Introduction **3**

2 Première démarche **3**
2.1 PPR 4
2.1.1 Description: 4
2.1.2 Design: 4
2.1.3 Fonctionnalités: 6
2.1.4 Besoins: 8
2.1.5 Installation: 8
2.2 CUPS 8
2.2.1 Description: 8
2.2.2 Design: 9
2.2.3 Fonctionnalités: 10
2.2.4 Besoins: 11
2.2.5 Installation: 11
2.3 Conclusion 11

3 Deuxième démarche **12**
3.1 Description 12
3.2 Design et Fonctionnement du programme 12
3.2.1 Description des classes 13
3.2.2 Description des Fichiers de Configuration 14
3.2.3 Description de la GUI: 18
3.3 Manuel d'utilisation 20
3.3.1 Utilisation Général: 20
3.3.2 Enlever/rajouter une imprimante 21
3.3.3 Modification du path de lp: 22
3.3.4 Ajouter/enlever une option 22
3.4 Problèmes rencontrés 23

4 Conclusion **25**

5 Remerciements **25**

6 Bibliographie **26**

A Code java **27**
A.1 Spooler.java 27
A.2 PrinterList.java 30
A.3 Printer.java 33

A.4 PrinterType.java 35
A.5 OptionsGroup.java 36
A.6 Options.java 40
A.7 Job.java 45
A.8 GuiFrame.java 49
A.9 PageCount.java 64

B Fichiers de Configuration **67**
B.1 hp2100 67
B.2 hp4100 70
B.3 lps17 73
B.4 lp Path 75

1 Introduction

Depuis la création d'UNIX, l'impression constitue une des limites majeures de ce système d'exploitation. Contrairement à d'autres systèmes d'exploitation comme Windows ou Mac OS, UNIX ne possède ni d'interface ni de système standard pour les imprimantes. Parmi les solutions existantes à ce problème, celle du groupe BSD (Berkeley) et System V sont les plus courantes. Cependant, la plupart des ces gestionnaires d'impression s'exécutent à partir d'une ligne de commande plutôt basique et peu conviviale. La configuration des imprimantes sur UNIX est plutôt compliquée et il serait plus commode de pouvoir choisir ses options d'impressions (imprimante, papier, mise en page) d'une manière graphique.

C'est dans ce contexte le Laboratoire de Systèmes d'Exploitations, disposant de plusieurs imprimantes postscripts dotées de capacités variées (tiroirs multiples, couleurs, transparents, recto/verso etc...), a édité ce projet pour remplacer l'ancien gestionnaire d'impression "TKLP" qui n'était plus adapté aux imprimantes récentes. Il s'agit donc de mettre en place un système interactif permettant de contrôler les diverses options des imprimantes.

2 Première démarche

La première démarche entreprise fut de faire une recherche sur une internet (www.sourceforge.net, www.freshmeat.net) pour trouver les projets existants allant dans le sens de celui du LSE. Nous avons trouvé plusieurs projets correspondant aux exigences, comme par exemple:

- PPR (http://ppr.trincoll.edu/)
- CUPS (http://www.cups.org/)
- Kprima (http://www.tuhlmann.purespace.de/projects/kprima/)
- LPRng (http://lprng.sourceforge.net/)
- pdq (http://pdq.sourceforge.net/)
- GNUlpr (http://lpr.sourceforge.net/)
- xprintutil (http://freshmeat.net/projects/xprintutil/)
- Lanzadera (http://www.maderassoriano.com/linux/)

Parmi tous ces projets, les deux premiers projets(PPR, CUPS) ont retenus particulièrement notre attention, et ceci pour plusieurs raisons; ils répondent à tous les besoins demandés, ils sont très complets et surtout, ils ont une documentation très détaillée, ce qui est très important dans l'optique de l'utilisation de l'un de ces projets. Pour le choix de l'un de ces deux projets, nous avons fait une description détaillée de chacun d'eux afin de déterminer leurs points forts et leurs points faibles respectifs.

2.1 PPR

2.1.1 Description:

PPR est un gestionnaire d'impression multi-plateformes (Solaris, SunOS, Linux) spécialement conçu pour fonctionner avec des imprimantes PostScript. Il est une alternative au programme "lp" utilisé sur les machine UNIX. En effet, PPR n'a nulle besoin des fonctions de lp, il possède lui-même un programme *ppr*, un démon *pprd*,etc...
PPR gère les différentes imprimantes soit via le protocol TCP/IP, soit en utilisant l'adresse IP. PPR peut être utilisé par des lignes de commande ou par une interface graphique WWW.

2.1.2 Design:

Tous les travaux d'impression passent par le programme "*ppr*". *ppr* lit le fichier PostScript à imprimer et le fractionne en trois fichiers. Ces trois fichiers sont placés dans un répertoire "jobs" par exemple. Un quatrième fichier est créé, celui-çi décrit le travail d'impression et ses différentes options d'impression. Dès que ces fichiers ont été créés, *ppr* envoie un message au démon (*pprd*) pour lui indiquer qu'une nouvelle impression est dans la file d'attente.

Lorsque *ppr* détermine qu'il est temps d'imprimer le document sur une imprimante, il appelle un programme appelé "*pprdrv*" qui lit les trois fichiers d'impression(comment, text, pages) et les rassemble dans un fichiers Postscript. Simultanément, il va lire les informations depuis le fichiers configuration de l'imprimante, le fichiers file d'attente (queueFile), et un fichier "PostScript Printer Description" (PPD). Il peut également inclure des caractères, ainsi que d'autres ressources dans le flux de données. Le programme *pprdrv* envoie le fichier à l'imprimante en appelant un programme d'interface d'imprimante. Le programme d'interface d'imprimante est responsable d'accepter l'impression, de la transmettre l'imprimante et de recevoir les messages d'erreurs, pour les

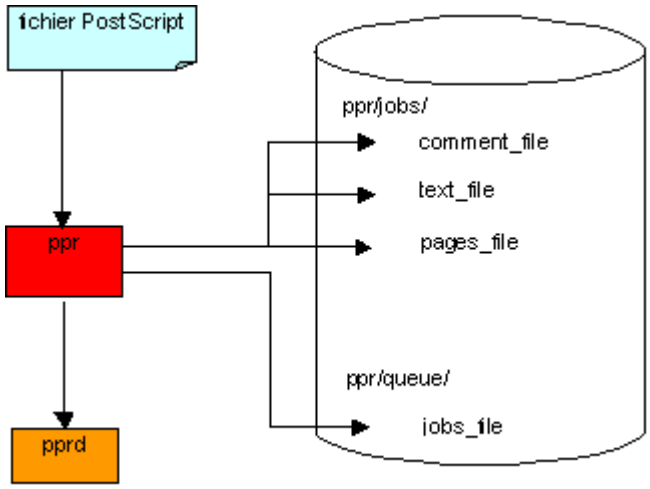

Figure 1: Une impression entre dans la file d'attente

transmettre ensuite au programme *pprdrv*. Dès que l'impression est terminée, le programme *pprdrv* se ferme, suivi du programme d'interface d'imprimante. Le démon d'impression, *pprd* le détecte et retire les impressions de la file d'attente et essaie d'informer l'utilisateur que l'impression est terminée.

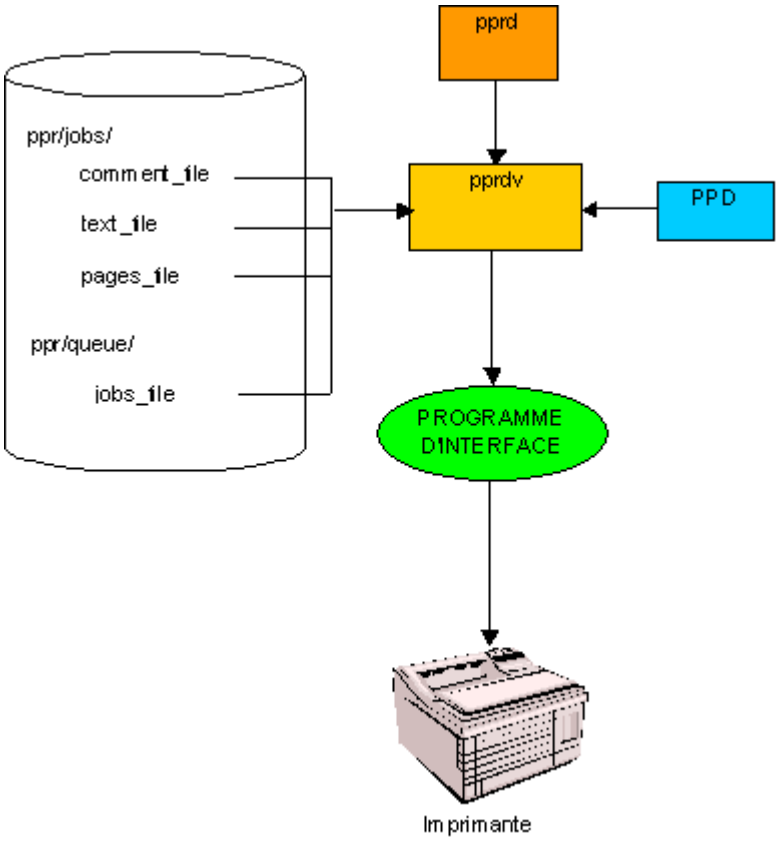

Figure 2: PPR lance une impression

Programme d'interface d'imprimante:
pprdrv communique avec l'imprimante à travers le programme d'interface d'imprimante. Ce programme lit le fichier PostScript depuis la stdin et l'envoie à l'imprimante. Si l'interface reçoit des messages depuis l'imprimante, il les copie sur la stdout.
L'interface indique également à l'imprimante où commence un fichier ps et où finit un autre.

2.1.3 Fonctionnalités:

Voici les différentes fonctionnalités que propose PPR:

1. "Options d'impression":
   - Choix du nombre de copies à imprimer d'un document.

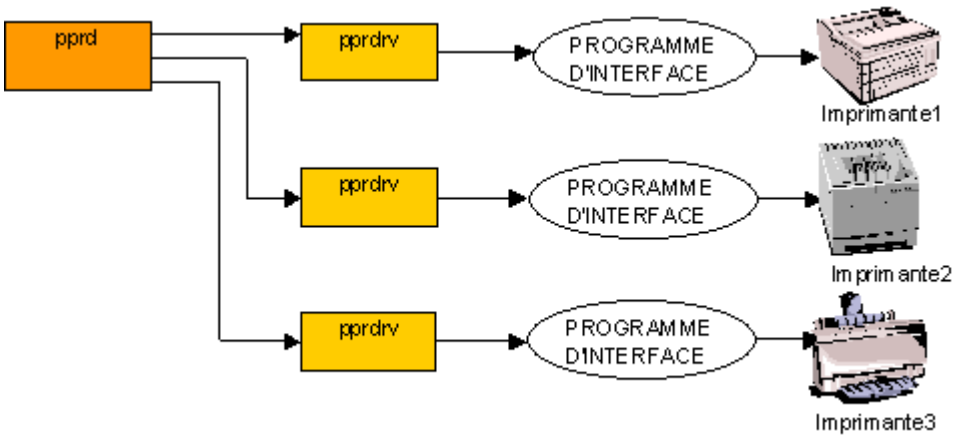

Figure 3: Plusieurs instance de *pprdrv* pour plusieurs imprimantes

- Choix du nombre de pages que l'on veut imprimer sur une.
- Imprimer un document en mode recto-verso ou en mode normale.
- Placer une signature sur chaque page.
- Choix de l'ordre d'impression du document : i.e. d'imprimer dans l'ordre croissant ou décroissant des pages.
- Imprimer une page de garde pour indiquer le nom de l'utilisateur qui a imprimé le document, ainsi que d'autres informations. Mais cette option peut être désactivée.

2. "Sélection automatique de tiroir":
   Il est possible de choisir dans quel tiroir de l'imprimante, on veut imprimer le document. Par contre, si aucun tiroir n'est sélectionné, PPR va choisir lui-même dans quel tiroir il va lancer l'impression par un système de sélection automatique.

3. "Groupes d'imprimantes"
   Des groupes d'imprimantes peuvent être créés. Ceci permet de lancer une impression sur un groupe d'imprimantes bien précises et PPR choisira sur laquelle de ces imprimantes il va lancer l'impression. A tout moment, une imprimante peut être inclue ou retirer d'un groupe.

4. "Messages"
   Un générateur de commentaire peut être activé. Il générera entre autres des messages indiquant la fin de l'impression, l'état de la file

d'attente, ainsi que des messages d'erreurs(arrêt, rejet ou annulation de l'impression).

5. "Filtres"
   PPR est capable d'utiliser des filtres pour convertir des fichiers au format PostScript. Les programmes supportés sont notamment: Troff, TeX, DVIPS, NetPBM, Acrobat Reader.

2.1.4 Besoins:

Pour installer le gestionnaire d'imprimante PPR, on a besoin des programmes suivants:

- Gunzip
- Tar
- Gcc version 2.5.8 ou plus rècente, Digital Unix 4.0
- GNU dbm (GDBM) 1.5 ou 1.7

2.1.5 Installation:

Lors de l'installation, il faut lancer la commande /.Configure. A ce moment, le programme s'installe dans les répertoires /etc/ppr, /usr/lib/ppr, /usr/share/ppr/ et /var/spool/ppr. Lorsque l'on lance la commande fixup, des utilisateurs "ppr" et "pprwww" et le groupe "ppr". Des liens symbolique sont également créés vers ppr, ppop, ppad et ppdoc dans le répertoire /usr/bin. Un script initial pour PPR est également installé dans le répertoire /etc/profile.d (si il existe), et configure Inetd ou Xinetd pour lancer le serveur web de mangement de PPR et le démon lpr. Dans cette configuration, le démon lpr de PPR va interférer avec le démon déjà existant.
Pour remédier à ce probléme, il faudrait installer PPR dans des répertoires différents que ceux pas défaut lors du lancement de la commande /.Configure.

2.2 CUPS

2.2.1 Description:

CUPS est un programme relativement nouveau (la première version date d'Octobre 1999), il a été développé par Easy Software Products dans le but de promouvoir un nouveau standard d'impression sur les plateformes Unix.
CUPS fonctionne sur quasiment toutes les distributions Unix/linux : AIX

/ Axon Linux / Caldera / Conectiva / Debian / easyLinux / FreeBSD / HP-UX / IRIX / Linux Mandrake / MIZI / Peanut / Red Hat / Solaris / SuSE / Tru64 UNIX / TurboLinux /.
CUPS ne comporte pas d'interface graphique(seule la version commerciale ESP Print Pro comprend une interface graphique), cependant il existe d'autres programmes, compatibles avec CUPS, qui viennent rajouter une interface graphique (QtCUPS, KUPS, etc...). Cependant, CUPS comporte un outil d'administration, accessible depuis un navigateur Internet lorsque CUPS est exécuté.

2.2.2 Design:

CUPS est basé sur le protocole IPP. IPP (Internet Printing Protocol) est une nouvelle technologie basée sur http, destinée à remplacer les systèmes propriétaires d'impression. Pour cela, IPP utilise une méthode simple consistant à envoyer les impressions d'un ordinateur à une imprimante connectée au réseau IP.
En développement depuis 1996, IPP a le support de plus de 25 entreprises, dont Microsoft, Novell, IBM, Xerox, Lexmark et Hewlett-Packard.
CUPS propose une interface moderne et complète pour le système UNIX. Il est adaptable pour de futures imprimantes, drivers ou protocoles tout en offrant la compatibilité avec les programmes UNIX existants.
Comme la plupart des autres systèmes d'impression, CUPS est construit autour d'un processus d'impression qui sépare les travaux d'impressions.
De manière simplifiée, on peut représenter le schéma de fonctionnement de CUPS comme cela.

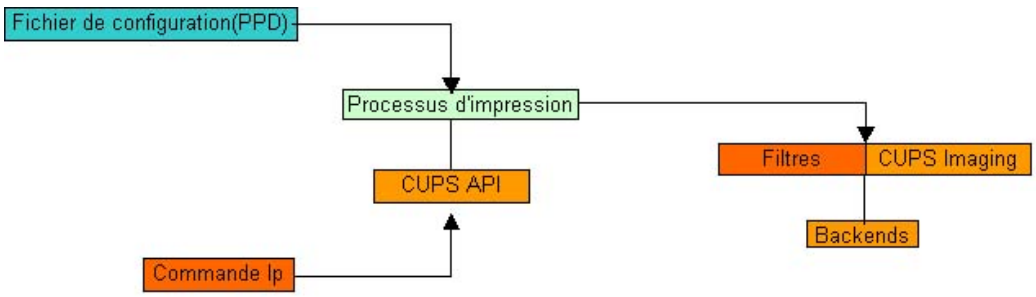

Figure 4: Schéma fonctionnel de CUPS

Processus d'impression

Le processus d'impression est une application serveur http/1.1 qui contrôle les requêtes. Il tient à jour une liste des imprimantes disponibles sur le LAN et sépare les travaux d'impressions en utilisant les filtres requis.

Fichier de configuration

Le fichier de configuration peut être par exemple un fichier PPD. Un fichier PPD est un fichier décrivant toutes les capacités d'une imprimante.

CUPS API

L'API CUPS contient toutes les fonctions nécessaires pour faire entrer les impressions dans la file d'attente, obtenir les informations de l'imprimante, manipuler les fichiers PPD.

Filtres et CUPS Imaging

Ces deux parties permettent de convertir, manipuler des fichiers de grande taille, réduire ou agrandir les images, sans effort supplémentaire pour l'utilisateur.

Backends

Le Backends réalise la tache finale, c'est-à-dire envoyer le document filtré à l'imprimante. Le Backends est aussi utilisé pour déterminer les imprimantes disponibles.

2.2.3 Fonctionnalités:

Comme CUPS utilise la commande lp, toutes les options d'impression de lp sont disponibles :

Options générales de mise en page:

- Choix de l'orientation
- Choix du type de papier
- Imprimer recto/verso
- Imprimer avec une page de garde
- Imprimer plusieurs pages sur une seule
- etc...

En plus des options de mise en page classiques, CUPS est capable de garder les travaux d'impression même après un redémarrage. Il inclut également un système d'authentification, de plus, chaque utilisateur possède ses préférences concernant le type de papier, mise en page, résolution.

2.2.4 Besoins:

- Un compilateur C et C++ doit être installé(par exemple gcc).

- Pour pouvoir utiliser les filtres, nous aurons besoin des librairies: JPEG, PNG, TIFF, ZLIB. CUPS fonctionnera sans ces filtres, mais les fonctionnalités seront réduites.

2.2.5 Installation:

Lancer les scripts suivants :

- Configure
  L'installation par défaut va mettre CUPS dans les répertoire /etc, /usr, /var, ce qui va effacer les commandes d'impression existantes. Rajouter –prefix=/some/directory pour changer de répertoire.

- make

- make install

  Pour lancer le programme:

- /usr/sbin/cupsd

2.3 Conclusion

Malgré le fait que que les deux projets précédemment cités soient complets et répondent totalement aux exigences, il nous a été impossible de les utiliser, et ceci pour pour plusieurs raisons; premièrement, ces programmes n'utilisent pas la commande "lp" sans la modifier (CUPS) ou ne l'utilisent pas du tout (PPR), et ceci s'est révéré être une exigence dans notre projet. Deuxièmement, ces deux programmes exigaient pour être installés que l'utilisateur soit "root", ce qui nous a pas été accordé par l'administrateur.

3 Deuxième démarche

3.1 Description

C'est pour toutes les raisons cits dans la conclusion, et après une nouvelle recherche sur internet(sur les sites www.sourceforge.net et www.freshmeat.net) pour trouver des projet qui se mettaient en "frontend" de lp, que nous avons décidé avec notre assistant de coder nous-mêmes le gestionnaires d'impression en Java.
Le programme se compose de 4 parties principales:

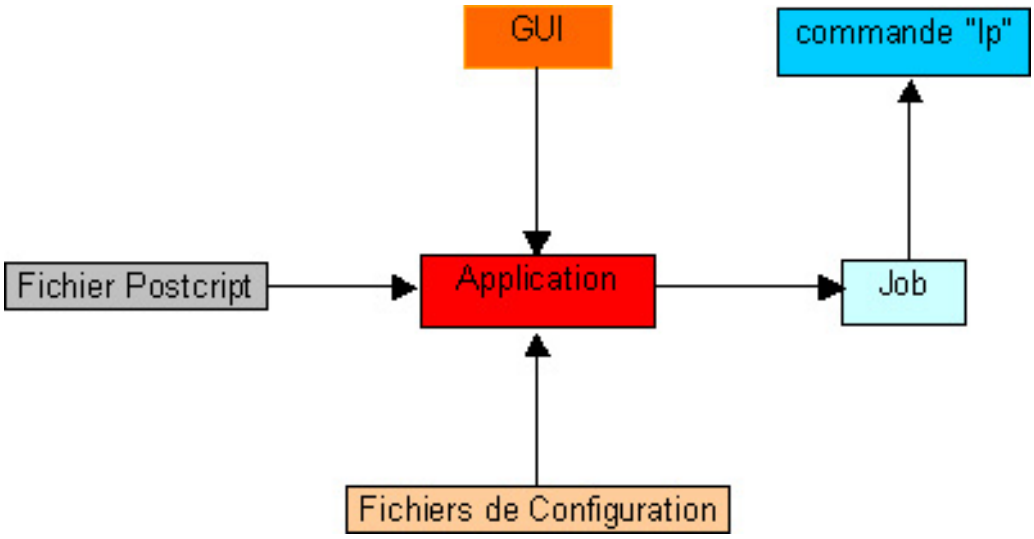

Figure 5: Schéma de fonctionnement général du programme

- L'interface Graphique qui permet à l'utilisateur d'interagir avec le programme.
- Les fichiers de configuration.
- L'application qui envoie les bons paramètres à *lp*.
- Le travail d'impression(Job).

3.2 Design et Fonctionnement du programme

Le corps du programme est divisé en cinq classes, décrites dans la section suivante, la hi'erarchie des classes est faite d'une manière récursive, de l'objet

le gén'eral au plus sp'ecifique. Lorsqu'une imprimante est selectionée par l'utilisateur, le programme détermine son nom, son type ou modèle et cherche ensuite la description de l'imprimante grâce aux fichiers de configuration. Les travaux d'impression sont ensuite envoyés à la classe Job qui se charge d'exécuter la commande *lp* avec les paramètres correctes. Une fois que les travaux d'impression ont été mis sur la file d'attente, le programme se ferme. La figure 6 montre l'organisation des classes.

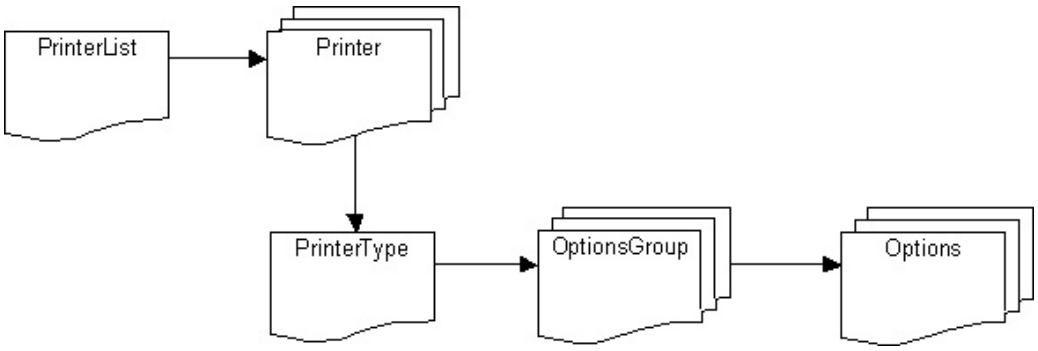

Figure 6: Hiérarchie des classes

3.2.1 Description des classes

PrinterList.java

PrinterList est une classe qui charge en mémoire la liste des imprimantes disponibles à partir d'un fichier de configuration (voir "Description des Fichiers de Configuration", printerList.dat). Elle peut également retrouver le type d'une imprimante à partir du nom de l'imprimante.

Printer.java

Dans le programme, une imprimante est définie par deux valeurs:

- son nom qui doit être unique, c'est l'identifiant de l'imprimante.
- son type ou modèle qui peut être adopté par d'autres imprimantes.

PrinterType.java

Cette classe va construire les options de l'imprimante à partir de son type ou modèle.

OptionsGroup.java
OptionsGroup.java charge en mémoire la description de l'imprimante à partir d'un fichier (voir "Description des Fichiers de Configuration", printerType.dat).

Options.java
Options.java charge en mémoire les commandes associées aux options de l'imprimante à partir d'un fichier (voir "Description des Fichiers de Configuration", commandType.dat). Cette classe retrouve également les commandes des options pour les envoyer en travaux d'impression.

Job.java
Tout fichier ou groupe de fichiers envoyés à l'imprimante est appelé un *job*. Job.java construit la ligne de commande qu'il va envoyer à *lp*, soit directement, soit à travers un *pipe*. L'exécution de *lp* se fait dans cette classe.

PageCount.java
Cette classe réunit quelques fonctions utiles telles que chercher le nombre de pages d'un document *ps* ou encore peut aller chercher le chemin la commande *lp*. Ceci se fait soit de manière directe ou à l'aide de commandes UNIX.

Spooler.java
Spooler.java contient la méthode main, elle lance l'interface graphique ainsi que la liste des imprimantes, cherche la commande *lp*. La méthode main peut prendre plusieurs arguments qui correspondent aux noms des fichiers à imprimer.
A noter que les noms des fichiers de configuration ou des noms de groupes peuvent être modifiés dans cette classe.

3.2.2 Description des Fichiers de Configuration

Les fichiers de configuration sont stockés dans des "Properties"(java.util) qui associe à chaques clés une valeur. Nous avons deux sortes de fichiers différents:
Le premier type de fichiers décrit la liste des imprimantes disponibles:

printerList.dat Ce fichier comporte 2 entrées, d'un cté le nom de l'imprimante, de l'autre son type ou modèle associé.

Syntaxe correcte: printerName=printerType Le deuxième type de fichiers est utilisé pour la decription d'une imprimante et de ses options disponibles, une imprimante est décrite par 2 fichiers:

printerType.dat décrit les capacités d'une imprimante, ses options sont ordonnées par groupes.
Syntaxe correcte: groupName=option(x),option(y),option(z)
Les noms de groupes supportés par le programme sont:

- Select_printer
- Number_of_copies
- No_banner
- Economy
- Orientation
- Duplex
- Media_size
- Media_type
- Media_source
- Media_output
- Pages_ranges
- Number_of_pages_per_sheet

commandType.dat est le fichier correspondant aux commandes *lp* associées aux options.
Syntaxe correcte: option=commande
Exemple d'un fichier **printerType.dat** (ici lps17.dat):

```
#Options of type lps17 printers
#
#Properties for Usual Options
Select_printer=choosePrinter
Number_of_copies=sheetcount
No_banner=No_banner
```

```
Economy=
#
#Properties for general Option
Orientation=Portrait,Landscape
Duplex=default,1-Sided,2-Sided
#
#Properties for media types
Media_size=default,A4,A5,A3,B5,B4,letter,ledger,legal,Executive,US10_envelope,DL_en
Media_type=default,Letterhead_paper,Drilled_paper,Transparencies,Gummed_labels
Media_source=default,Upper_input_tray,Bottom_input_tray,Large_capacity_input_tray,M
Media_output=default,Side_tray(face_up),Lower_tray(face_down),Mail_box,Side_tray,To
#
#Properties for document options
Page_ranges=All,From_to
Number_of_pages_per_sheet=default,One_per_sheet,Two_per_sheet,Four_per_sheet
```

Exemple d'un fichier **commandType.dat** (ici commandlps17.dat):

```
#Command List for printers of type lps17
#
default=
#INTRAY
Upper_input_tray=-o intray\=top
Bottom_input_tray=-o intray\=bottom
Large_capacity_input_tray=-o intray\=lcit
Manual_feed_slot=-o intray\=manual
Envelope_tray=-o intray\=envelope
#
#NUP
One_per_sheet=-o nup=1
Two_per_sheet=-o nup=2
Four_per_sheet=-o nup=4
#
#OUTTRAY
Side_tray(face_up)=-o outtray\=face-up
Lower_tray(face_down)=-o outtray\=lower
Mail_box=-o outtray\=mb
Side_tray=-o outtray\=side
Top_tray=-o outtray\=top
Upper_tray=-o outtray\=upper
#
#SHEETCOUNT
sheetcount=-o sheetcount\=
#
#MEDIA SIZE
ISO_A4=-o sheetsize\=a4
ISO_A5=-o sheetsize\=a5
ISO_A3=-o sheetsize\=a3
ISO_B5=-o sheetsize\=b5
ISO_B4=-o sheetsize\=b4
US_letter=-o sheetsize\=letter
US_ledger=-o sheetsize\=ledger
US_legal=-o sheetsize\=legal
Executive=-o sheetsize\=executive
US10_envelope=-o sheetsize\=business_envelope
ISO_DL_envelope=-o sheetsize\=dl
7_envelope=-o sheetsize\=7_envelope
```

```
#
#SIMPLEX AND DUPLEX
1-Sided=-o simplex
2-Sided=-o duplex
#
#ORIENTATION
Portrait=-o orient\=port
Landscape=-o orient\=land
#
#SPECIAL MEDIA
Letterhead_paper=
Drilled_paper=
Transparencies=-o face-up
Gummed_labels=-o face-up
#
#
choosePrinter=-d
Odd=-o page-set\=odd
Even=-o page-set\=even
From_to=-P
No_banner=-o nobanner
```

A noter:

- Les noms de groupes dans le fichier printerType.dat doivent être tous présent et doivent avoir les noms prédéfinis plus haut.
- Les options à droite dans le fichier printerType.dat et à gauche dans le fichier commandType.dat doivent avoir le même nom, sinon le système ne les reconnaîtra pas.
- Si la commande comporte est signe "=", il faut qu'il soit précédé d'un \ (ex:Portrait=-o orient \=**port)**
- **Un groupe d'options ou une commande pour une option peuvent comporter une valeur nulle.**

3.2.3 Description de la GUI:

L'interface graphique de ce programme a été réalisée à partir du package awt de Java, et ceci pour la rendre la plus compatible possible dans l'optique d'une utilisation multiplateforme (UNIX, MAC). Notre but était de faire une interface graphique à la fois robuste et conviviale.

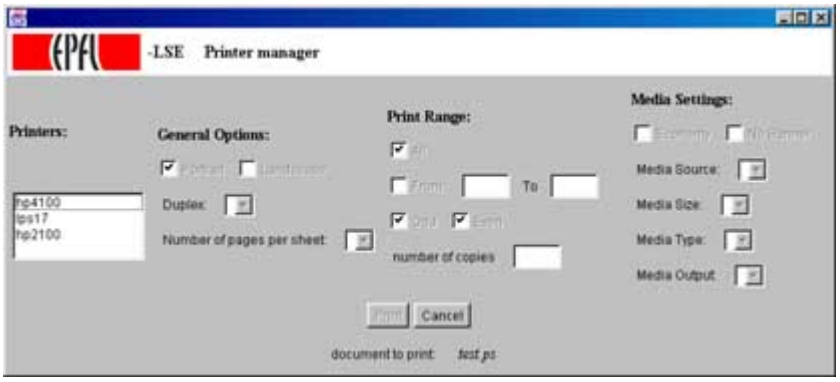

Figure 7: Interface Graphique du Gestionnaire d'impression

La GUI est structurée de la manière suivante; un borderLayout sépare la fenêtre en quatre parties (partie nord, partie ouest, partie centrale et partie sud). La partie nord (Panel panelNorth) est composée du titre et du logo de l'EPFL, la partie ouest Panel panelWest) est composée d'une liste comportant toutes les imprimantes gérées par le gestionnaire d'impression, La partie sud (Panel panelSouth) est composée des deux boutons "Print" et "Cancel" ainsi que d'un Label indiquant le(s) document(s) à imprimer, enfin la partie centrale (Panel panelCentral) regroupe toutes les options gérant l'impression. Cette dernière partie fera l'objet plus loin d'une description plus approfondie.

La partie central, gérée par un FlowLayout, est composé de trois panneaux (Panel panelGeneral, Panel panelPrintRange et Panel panelMedia), qui représentent chacun un type d'options d'impression. Le premier panneau regroupe les options générales (orientation, recto-verso, etc.), le deuxième, les options permettant de selectionner la partie à imprimer (nombre de copies, page paire/impaire, etc.) et le troisième, les options permettant de contrôler l'imprimante elle-même (mode economy/normal, tiroir d'entrée et tiroir de sortie, etc.).

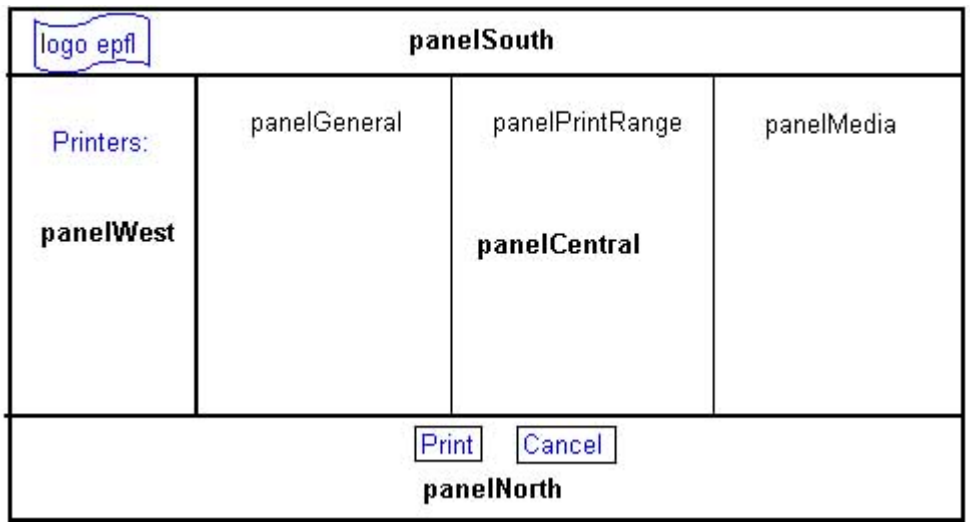

Figure 8: Structure de l'interface graphique

3.3 Manuel d'utilisation

Ce manuel décrit les principales fonctions de ce programme, ainsi que des possibilités de modification des options, du rajout ou de la suppression d'une imprimante.

3.3.1 Utilisation Général:

Installation et execution du programme

Pour installer le programme, il suffit de copier le fichier "projet.jar" sur son ordinateur.

Pour l'executer (i.e pour lancer une impression), il faut taper dans le terminal la commande suivante:

java -cp projet.jar ch.epfl.lse.impression.Spooler filename_to_print

Il est possible d'imprimer plusieurs documents avec les mêmes options, pour cela il faut taper la commande:

java -cp projet.jar ch.epfl.lse.impression.Spooler filename1_to_print filename2_to_print filename3_to_print etc.

Lancer une impression

Pour lancer une impresion, il faut effectuer les opérations suivantes:

- ex'ecuter le programme
- Lorsque que la fenêtre de l'interface graphique est ouverte, selectionner une imprimante dans la liste tout à gauche de la fenêtre.
- Toutes les options par défaut de l'imprimante séléctionnée vont s'afficher
- Choisir les différentes options d'impression voulues
- Cliquer sur le bouton "Print" pour lancer l'impression
- Pour fermer la fenêtre sans imprimer, cliquer sur le bouton "Cancel"

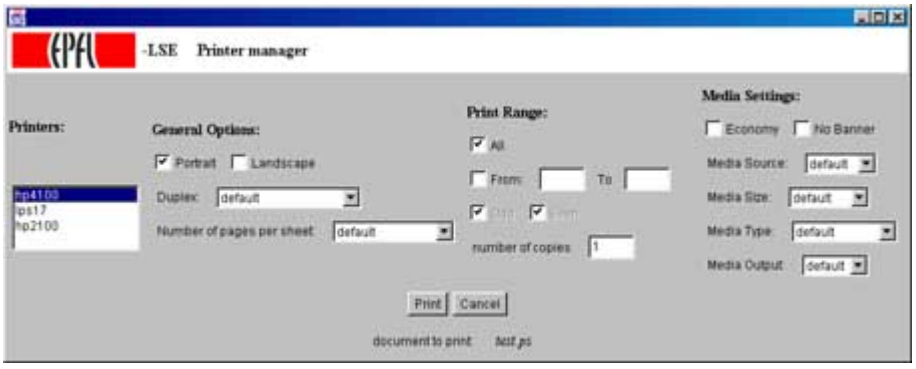

Figure 9: Interface graphique du programme après selection de l'imprimante

3.3.2 Enlever/rajouter une imprimante

L'opération d'enlever/rajouter une imprimante ne nécessite que des modifications au niveau des fichier de configuration.

Enlever une imprimante:

Pour supprimer une imprimante, il faut supprimer dans le fichiers "printerList.dat" la ligne correspondante à l'imprimante à supprimer. Il faut également supprimer les fichiers "printer_type.dat" et "commandprinter_type.dat" (par exemple, "hp2100.dat" et "commandhp2100.dat") **seulement si** aucune imprimante du même type n'est utilis'ee. Cette dernière opération n'est pas nécessaire au bon fonctionnement du programme, mais les deux fichiers précités à supprimer ne sont plus utiles et sont donc à supprimer.

Rajouter une imprimante:
Pour rajouter une imprimante, deux cas se présentent:

- Il y a déjà une imprimante du même type (avec les mêmes options) dans le laboratoire; et là il suffit de rajouter une ligne dans le fichier "printerList.dat" avec le nom de l'imprimante et le type de l'imprimante (le même que celle de même type déjà présente dans le laboratoire) séparé par un espace. Par exemple, si on veut rajouter l'imprimante "hp6100" et qu'elle est du même type que l'imprimante "hp2100", il s'agit d'ajouter la ligne suivante dans le fichier "printerList.dat":

  hp6100 hp2100

- Aucune imprimante correspond au même type que l'imprimante à rajouter; là, cette fois çi, il faut rajouter une ligne dans le fichier "printerList.dat" avec le nom de l'imprimante et le type de l'imprimante à rajouter (comme on l'a déjà fait dans le cas précédent, mais cette fois-ci, le type de l'imprimante est nouveau). Ensuite, il faut créer deux fichiers (printer_type.dat et commandprinter_type.dat) en suivant le modèle exposé au chapitre 3.2.2.

3.3.3 Modification du path de lp:

Pour modifier le path de *lp*, il suffit de modifier le fichier "lpPath.dat" et d'inscrire le nouveau path de *lp* à la place de l'ancien. En effet, dans ce fichier, la seule ligne présente est le path de lp (par exemple: /usr/bin/lp).

Si, à l'ouverture du programme, le message *"Warning: lpPath.dat may not contain the correct path for lp"* apparait sur le terminal, cela veut dire que le path de lp est mal configuré. Il s'agit dans ce cas de modifier ce path dans le fichiers "lpPath.dat".

3.3.4 Ajouter/enlever une option

L'opération d'enlever/ajouter une option d'impression nécessite des modifications dans les fichiers de configuration mais aussi dans certain cas dans le code lui-même:

Enlever une option:
Pour enlever une option d'impression à une imprimante, il suffit de modifier les deux fichiers de configurtion correspondant à l'imprimante. En effet, dans le fichier "printer_type.dat", il faut supprimer l'option à enlever. Puis, dans le

fichier “commandprinter_type.dat, il faut supprimer la ligne correspondant à cette option. En ce qui concerne l'interface graphique; si l'option à supprimer était représentée par un bouton, alors le bouton sera désactivé, et si l'option faisait partie d'une liste, alors l'option sera juste enlevée de la liste.

Ajouter une option:
Pour ajouter une option d'impresion à une imprimante, deux cas se présentent:

- **L'option à ajouter est déjà représentée dans l'interface graphique par un bouton ou doit juste être ajoutée à une liste:**
  Dans ce cas, des modifications dans le fichiers de configurations suffisent; en effet, il suffit de rajouter dans le fichier “printer_type.dat” l'option désirée en suivant la syntaxe de ce fichier (chapitre 3.2.2), si l'option fait partie d'une liste déjà existante dans l'interface graphique, il faut rajouter cette option dans la ligne représentant cette liste. Il faut également rajouter l'option et sa commande dans le fichier “commandprinter_type.dat” selon la syntaxe de ce fichier (chapitre 3.2.2).

- **L'option à ajouter n'est pas présente dans l'interface graphique ni sous forme de bouton, ni sous forme de liste:**
  Dans ce cas, il faut faire les mêmes opérations que dans le cas précédent. Mais, il faut faire en plus des modification dans le code. En effet, il faut créer dans l'interface graphique un élément (bouton ou liste) qui représente l'option à ajouter. Ensuite, il faut créer une variable dans la classe “job.java” qui représente l'état de l'option (activé/désactivé ou sélectionn'e/non-sélectionn'e) et lui affecter dans la classe “GuiFrame” la bonne valeur, et ceci lorsque l'utilisateur clique sur le bouton “Print”. Il faut également modifier la méthode “sendToJob()” dans la classe “Options”, afin que cette méthode ait chercher l'état de la variable, représentant l'option à ajouter, pour la rajouter à la ligne de commande.

3.4 Problèmes rencontrés

Les problémes que nous avons rencontré étaient surtout liés aux imprimantes et/où au fonctionnement parfois instable de *lp* pour certaines options. En effet, certaines options comme le *page range* refusent de fonctionner. Il faut parfois les remplacer par le *page range* spécifique de l'imprimante en question. Nous avons également rajouté une option *default* pour toutes les options disponibles dans l'interface graphique, car nous avons constaté que lorsque l'on envoyait trop d'arguments à l'imprimante, des plantages pouvaient survenir, selon le fichier à imprimer. L'option *default* est rajoutée devant toutes

les options dans le fichier printerType.dat et nous laissons un champ vide pour la commande dans le fichier commandType.dat (*default=*). Ainsi, la ligne de commande pour *lp* contient un minimum d'arguments.

4 Conclusion

Les objectifs du projet nous semblent atteints, en effet, le programme gérant l'infrastructure d'impression fonctionne. Il est facilement configurable grâce aux fichiers de configuration simples et en mode texte. De plus, l'utilisation de ce programme est assez conviviale.

Les améliorations possibles du projet pourraient être les suivantes:

- Une interface graphique plus modulable facilitant le rajout/suppression de nouvelles options.
- L'intégration des fichiers PPD(Poscript Printer Description), qui permettraient de connaitre les options disponibles de l'imprimante et ainsi faciliter l'intégration de nouvelles imprimantes.

Ce projet nous a donné l'opportunité de développer une application réelle et complète, et ceci d'une manière relativement autonome. Ce projet nous a également permis de consolider nos connaissances relatives au processus lp et aux processus d'impressions en général. Il nous a également permis d'élargir nos connaissances en Java, par exemple en ce qui concerne les "exec", "getRessource", pipe et Flux de données, afin de coder une application réelle robuste.
Nous nous sommes également familiariser avec l'outil fort intéressant qu'est "LaTex" pour la création du rapport.

5 Remerciements

Nous voulons remercier le laboratoire LSE pour nour avoir donner la possibilité de coder une application réelle qui sera, nous l'espérons, utile dans le futur, et en particulier le Professeur A. Schiper et M. Wiesmann. Un grand merci à M. Wiesmann pour nous avoir aider à structurer notre travail et pour les connaissances nouvelles qu'il nous a apportées.

6 Bibliographie

- La Mailing List du projet PPR et en particulier David Chappell (http://ppr.trincoll.edu)
- La Mailing List du Projet CUPS (http://www.cups.org)
- Le site (http://www.linuxprinting.org)
- JavaTM 2 Platform, Standard Edition, v 1.3.1, API Specification (http://lamscours-java.epfl.ch/2001/docs/java/docs/api/index.html)

A Code java

A.1 Spooler.java

```
package ch.epfl.lse.impression;

import java.util.*;
import java.io.*;
import java.text.*;

/** Classe principale qui lance le programme, contient la mthode main */
public class Spooler {

    static PrinterList printerList = new PrinterList();
    static Job job = new Job();
    static OptionsGroup optionsGroup = new OptionsGroup();
    static Printer printer = new Printer();
    static PageCount pageCount = new PageCount(job);
    static GuiFrame f;
    /** Fichier printerList.dat */
    static final String printerListDat = "printerList.dat";
    /** Fichier commandType.dat */
    static final String commandDat = "command";
    /** Fichier preferences.dat */
    static final String preferences = "preferences.dat";
    /** Fichier lpPath.dat */
    static final String lpPath = "lpPath.dat";
    /** nom du groupe choosePrinter dans le fichier type.dat */
    static final String choosePrinter = "choosePrinter";
    /** nom de l'option landscape dans le fichier type.dat */
    static final String checkboxLandscape = "Landscape";
    /** nom du l'option dans le fichier type.dat */
    static final String checkboxPortrait = "Portrait";
    /** nom du groupe duplex dans le fichier type.dat */
    static final String choiceDuplex = "Duplex";
    /** nom du groupe number of pages per sheet dans le fichier type.dat */
    static final String choiceNumberPages = "Number_of_pages_per_sheet";
    /** nom du groupe media source dans le fichier type.dat */
    static final String choiceMediaSource = "Media_source";
    /** nom du groupe media size dans le fichier type.dat */
    static final String choiceMediaSize = "Media_size";
```

```
    /** nom du groupe media type dans le fichier type.dat */
    static final String choiceMediaType = "Media_type";
    /** nom du groupe media output dans le fichier type.dat */
    static final String choiceMediaOutput = "Media_output";
    /** nom du groupe orientation dans le fichier type.dat */
    static final String orientation = "Orientation";
    /** nom de l'option from...to dans le fichier type.dat */
    static final String checkboxFrom = "From_to";
    /** nom de l'option odd dans le fichier type.dat */
    static final String checkboxOdd = "Odd";
    /** nom de l'option even dans le fichier type.dat */
    static final String checkboxEven = "Even";
    /** nom de l'option multiples copies dans le fichier type.dat */
    static final String multipleCopies = "MultipleCopies";
    /** nom de l'option from...to dans le fichier type.dat */
    static final String printFromTo = "From_to";
    /** nom du groupe economy dans le fichier type.dat */
    static final String checkboxEconomy = "Economy";
    /** nom du groupe no banner dans le fichier type.dat */
    static final String checkboxNoBanner = "No_banner";
    /** nom du groupe page range dans le fichier type.dat */
    static final String pageRange = "Page_ranges";



    public Spooler() {
    }
    /**
     * Mthode main qui lance l'interface graphique, charge la liste des imprimantes
     * Le mthode peut prendre plusieurs arguments en ligne de commande, qui corresp
     * aux fichiers  imprimer.
     */
    public static void main(String arguments[]) {

for (int i = 0; i < arguments.length; i++) {
    job.filename.add(arguments[i]);
}

for(int i=0; i < job.filename.size(); i++)
job.files = (String)job.filename.get(0);
```

```
for (int i = 1; i < job.filename.size(); i++) {
    job.files = job.files + ", " +job.filename.get(i);
}


printerList.loadPrinterList();
printerList.getPrinterByName();

Collator cl = Collator.getInstance();
job.job = pageCount.loadlpPath();
if(cl.equals(pageCount.getlpPath(),pageCount.loadlpPath()) == false) {
    System.out.println("Warning: lpPath.dat may not contain the correct path for lp
    System.out.println("which response for lp is "+pageCount.getlpPath());
    System.out.println("lpPath.dat contains " + pageCount.loadlpPath());
}


f = new GuiFrame(job);
f.setVisible(true);
f.pack();

    }

}
```

A.2 PrinterList.java

```
package ch.epfl.lse.impression;

import java.util.*;
import java.io.*;
/**
 *PrinterList est une classe qui permet de charger la liste des imprimantes
 */
class PrinterList {

    Spooler spooler = new Spooler();
    Printer printer = new Printer();
    /** Table de Hash des noms et types des imprimantes */
    static Properties printerList = new Properties();
    /** Liste des nom des imprimantes */
    static ArrayList printerNameList = new ArrayList();
    Enumeration e;



    /**
     * Charge la liste des imprimantes  partir d'un fichier(printerlist.dat).
     */

    void loadPrinterList() {
try {
            Class pl = this.getClass();
            InputStream resource_stream = pl.getResourceAsStream(spooler.printerLis
    printerList.load(resource_stream);
} catch(FileNotFoundException e) {
    System.out.println("File: "+spooler.printerListDat+ " not found");
} catch(IOException e){
e.printStackTrace();
} // try / catch
    }


    /**
     * Cherche le type d'une imprimante  partir de son nom.
```

```
     * @param printerName Le nom de l'imprimante
     * @return Le type de l'imprimante
     */

String getPrinterType(String printerName) {

      String type = printerList.getProperty(printerName);
      return type;
    }


     /**
 * Charge tous les noms des imprimantes dans une liste.
 * @return la liste contenant le nom de toutes les imprimantes
 */
    ArrayList getPrinterByName() {


      e = printerList.propertyNames();
      while (e.hasMoreElements()) {
          String printerName = (String)e.nextElement();
          printerNameList.add(printerName);
          }
      return printerNameList;
    }


     /**
 * Construit une imprimante(groupes d'options et options).
 * @param printerName Le nom de l'imprimante
 */
    void createPrinter(String printerName) {

String printerType = getPrinterType(printerName);
printer.buildPrinter(printerName,printerType);
    }


     /**
 * Charge une imprimante depuis le fichier preferences.dat
 * (Voir aussi savePrinter).
```

```
 * Mthode non utilise pour le moment.
 */
    void loadPrinter() {

         String printerName = "";
try {
FileReader file = new FileReader(spooler.preferences);

int c;
do {
    c = file.read();
    if(c>0)
            {printerName += ((char)c);}
    }while (c>0);

        file.close();
        createPrinter(printerName);
}
catch(IOException e) {
    System.out.println("Unable to read from file " +spooler.preferences);
}
    }

}
```

A.3 Printer.java

```
package ch.epfl.lse.impression;

import java.util.*;
import java.io.*;
/**
 * Printer est une classe permettant de construire les imprimantes  partir de leur
 * de sauver ou charger une imprimante.
 */
class Printer {
    PrinterType printerType = new PrinterType();


    /** Construit une imprimante(groupes d'options et options).
     * @param printerName Le nom de l'imprimante
     * @param type Le type de l'imprimante
     */
    void buildPrinter(String printerName, String type) {

printerType.buildPrinterType(type);
    }
     /** Sauve l'imprimante dans un fichier. Cette mthode peut tre utilis
 * pour mettre une imprimante par dfaut. (Voir aussi loadPrinter).
 * @param printerName Le nom de l'imprimante
 */
    void savePrinter(String printerName) {


try {
    FileWriter fw = new FileWriter("preferences.dat");
    fw.write(printerName);
    fw.close();
}
catch(IOException e) {
    System.out.println("Error while writing in preferences.dat");
}
    }
```

```
}
```

A.4 PrinterType.java

```
package ch.epfl.lse.impression;

import java.util.*;
import java.io.*;

/**
 * Classe relative au type de l'imprimante, permet de construire le type d'une impr
 */
class PrinterType {
    OptionsGroup optionsGroup = new OptionsGroup();
    Options options = new Options();

     /**
 * Construit une imprimante  partir de son type uniquement.
 * @param type Le type de l'imprimante
 */
    void buildPrinterType(String type) {

        optionsGroup.buildOptions(type);
        options.buildCommand(type);
    }

}
```

A.5 OptionsGroup.java

```
package ch.epfl.lse.impression;

import java.util.*;
import java.io.*;
/**
 * OptionsGroup contient toutes les options d'une imprimante, ordonnes par groupes.
 * par ex: Orientation=Portrait,Landscape
 */
class OptionsGroup {

    Options options = new Options();
    /** Table de Hash qui associe les noms de groupes aux options */
    static Properties optionsGroup = new Properties();
    /** Table de Hash qui associe une option  un nom de groupe */
    static Properties tokenizedOptionsGroup = new Properties();
    /** Liste contenant les options d'un groupe */
    static ArrayList optionsList = new ArrayList();
    Enumeration e;


/**
 * Charge la description d'une imprimante(ses options classes en groupes) dans
 * une Properties. La cl correspond au groupe et la valeur correspond  toutes les o
 * @param type Le type de l'imprimante
 */


    Properties buildOptions(String type) {

try {
            Class pl = this.getClass();
            InputStream fis = pl.getResourceAsStream(type+".dat");
    optionsGroup.load(fis);
}
catch(FileNotFoundException e) {
    System.out.println("File: "+type+".dat not found");
}
```

```
catch(IOException e){e.printStackTrace();}

        buildTokenizedOptionsGroup();

        return optionsGroup;

    }
     /**
 * Charge les options de l'imprimante dans une Properties, mais dans l'ordre invers
 * La cl correspond  l'option et la valeur correspond au groupe.
 * Utile pour savoir  quel groupe appartient une option.
 * @param type Le type de l'imprimante
 */

    Properties buildTokenizedOptionsGroup() {

         e = optionsGroup.propertyNames();
         while (e.hasMoreElements()) {
             String typeName = (String)e.nextElement();
             StringTokenizer st = new StringTokenizer(optionsGroup.getProperty(type

             while (st.hasMoreTokens()) {
              tokenizedOptionsGroup.setProperty(st.nextToken(),typeName);
              }
          }

    return tokenizedOptionsGroup;
    }
     /**
 * Mets toutes les options d'un groupe dans une liste.
 * (Voir aussi clearListOptions)
 * @param groupName Le nom du groupe d'options
 * @return La liste des options du groupe d'options
 */
    ArrayList listOptions(String groupName) {


String opt = optionsGroup.getProperty(groupName);
StringTokenizer st = new StringTokenizer(opt,",");

while (st.hasMoreTokens()) {
```

```
    String list = (String)st.nextToken();
    optionsList.add(list);
}

return optionsList;
    }
     /**
 * Efface la liste des options du groupe
 */
    void clearListOptions() {

optionsList.clear();
    }

     /**
 * Permets de retrouver toutes les options d'un groupe d'options.
 * @param optionGroupName Le nom du groupe d'options
 * @return La liste des options du groupe d'options
 */
    void getOptions(String optionGroupName) {

Enumeration t = tokenizedOptionsGroup.keys();
while (t.hasMoreElements()) {
    System.out.println(t.nextElement());
}

    }
     /**
 * Permets de savoir  quel groupe d'options appartient une option.
 * @param option Le nom de l'option
 * @return Le groupe d'options de l'option
 */
    String getGroup(String option) {

String group = tokenizedOptionsGroup.getProperty(option);

return group;
    }
     /**
 * Affiche dans le terminal quels sont les commandes manquantes.
 */
```

```
    void verifyCommand() {

Enumeration t = tokenizedOptionsGroup.keys();
while (t.hasMoreElements()) {
    String e = (String)t.nextElement();
    String comp = options.getCommand(e);
    if (comp == null)
System.out.println("Missing Command for " +e);
}
    }



}
```

A.6 Options.java

```
package ch.epfl.lse.impression;

import java.util.*;
import java.io.*;
import java.text.*;
/**
 * Options contient les commandes d'une imprimante. Cette classe envoie galement
 * les bonnes commandes  Job.
 */
class Options {

    Job job;
    Spooler spooler = new Spooler();
    static Properties commandList = new Properties();
    String optionCommand;
    Collator cl = Collator.getInstance();
    String nbCopies;
    String printFromTo;
    String line;

    Options(){}

    Options(Job job){
this.job = job;
    }
    /**
     * Charge toutes les commandes des options d'une imprimante  partir d'un fichie
     * @param type Le type de l'imprimante
     */
    Properties buildCommand(String type) {


        try {
            Class pl = this.getClass();
            InputStream fis = pl.getResourceAsStream(spooler.commandDat+type+".dat"
    commandList.load(fis);
}
catch(FileNotFoundException e) {
```

```
    System.out.println("File: " + spooler.commandDat+type+".dat not found");
}
catch(IOException e){e.printStackTrace(); }


        return commandList;
    }
    /**
     * Permets de retrouver la commande d'une option.
     * @param optionName Le nom de l'option
     * @return La commande de l'option
     */
    String getCommand(String optionName) {

String command = commandList.getProperty(optionName);
return command;
    }
    /**
     * Cherche toutes les commandes des options et les envoies  Job.
     */
    void sendToJob() {
ArrayList buf = new ArrayList();

if(job.printerName != null)
    buf.add(choosePrinter(job.printerName));

if(job.orientation != null && getCommand(job.orientation) != null)
    buf.add(getCommand(job.orientation));

if(job.duplex != null && getCommand(job.duplex) != null)
    buf.add(getCommand(job.duplex));

if(job.numberPages != null && getCommand(job.numberPages) != null)
    buf.add(getCommand(job.numberPages));

if(job.mediaSource != null && getCommand(job.mediaSource) != null)
    buf.add(getCommand(job.mediaSource));

if(job.mediaOutput != null && getCommand(job.mediaOutput) !=  null)
    buf.add(getCommand(job.mediaOutput));
```

```
if(job.mediaSize != null && getCommand(job.mediaSize) != null)
    buf.add(getCommand(job.mediaSize));

if(job.mediaType != null && getCommand(job.mediaType) != null)
    buf.add(getCommand(job.mediaType));

if(job.numberCopies != null && multipleCopies() != null)
    buf.add(multipleCopies());

if(job.range && job.printFrom!=null && job.printTo != null && printFromTo() != null
    buf.add(printFromTo());

if(job.odd && !job.even)
    buf.add(getCommand(spooler.checkboxOdd));

if(job.even && !job.odd)
    buf.add(getCommand(spooler.checkboxEven));

if(job.economy  && getCommand(spooler.checkboxEconomy)!= null)
    buf.add(getCommand(spooler.checkboxEconomy));

if(job.noBanner && getCommand(spooler.checkboxNoBanner) != null)
    buf.add(getCommand(spooler.checkboxNoBanner));


for(int i = 0; i<buf.size();i++) {
    job.buildLine((String)buf.get(i));
}

job.print();
job.job = "lp";
buf.clear();
    }
     /**
 * Cherche la commande pour l'option choosePrinter et rajoute l'imprimante
 *  la ligne de commande.
 * @param printerName Le nom de l'imprimante
 * @return La commande pour choosePrinter
 */
    String choosePrinter(String printerName) {
```

```
if(getCommand(spooler.choosePrinter) != null) {
    String command = getCommand(spooler.choosePrinter);
    line = command + " " + printerName;
}
return line;
    }
```

```
     /**
 * Cherche la commande pour multipleCopies et lui ajoute les bons paramtres
 * @return La commande pour multipleCopies
 */
    String multipleCopies(){


if (job.numberCopies != null && cl.equals(job.numberCopies, "1") == false) {
    String command = getCommand(spooler.multipleCopies);
    if(command != null)
    nbCopies = command + " " + job.numberCopies;
}
return nbCopies;
    }
     /**
 * Cherche la commande pour printFromTo et lui ajoute les bons paramtres
 * @return la commande pour printFromTo
 */
    String printFromTo() {

if (job.printFrom!=null && job.printTo != null) {
    String command = getCommand(spooler.printFromTo);
    if(command != null)
    printFromTo = command +" "+job.printFrom+ "-" +job.printTo;
}
return printFromTo;
    }
```

```
}
```

A.7 Job.java

```
package ch.epfl.lse.impression;

import java.lang.*;
import java.util.*;
import java.io.*;

/**
* Job est la classe qui permet d'excuter une ligne de commande donne.
* (lp avec les bons paramtres).
*/
class Job {
    /** rpertoire dans lequel se trouve lp */
    public final String lp_command = "/usr/bin/lp";
    Spooler spooler = new Spooler();
    /** commande lp */
    String job = lp_command;
    /** Liste des fichiers  imprimer */
    ArrayList filename = new ArrayList();
    /** String contenant tous les fichiers  imprimer */
    String files;


    /** Variable modifie par la GUI pour le checkbox printAll */
    String printAll;
    /** Variable modifie par la GUI pour le checkbox economy */
    boolean economy;
    /** Variable modifie par la GUI pour le checkbox noBanner */
    boolean noBanner;
    /** Variable modifie par la GUI pour le checkbox odd */
    boolean odd;
    /** Variable modifie par la GUI pour le checkbox even */
    boolean even;
    /** Variable modifie par la GUI pour le checkbox orientation */
    String orientation;
    /** Variable modifie par la GUI pour le choice duplex */
    String duplex;
    /** Variable modifie par la GUI pour le textfield numberPages */
    String numberPages;
```

```
    /** Variable modifie par la GUI pour le checkbox range */
    boolean range;
    /** Variable modifie par la GUI pour le textfield printFrom */
    String printFrom;
    /** Variable modifie par la GUI pour le checkbox printTo */
    String printTo;
    /** Variable modifie par la GUI pour le textfield numberCopies */
    String numberCopies;
    /** Variable modifie par la GUI pour le choice mediaSource */
    String mediaSource;
    /** Variable modifie par la GUI pour le choice mediaSize */
    String mediaSize;
    /** Variable modifie par la GUI pour le choice mediaType */
    String mediaType;
    /** Variable modifie par la GUI pour la liste printerName */
    String printerName;
    /** Variable modifie par la GUI pour le choice mediaOutput */
    String mediaOutput;


    /**
     * Construit la ligne de commande pour lp
     * @return La ligne de commande
     */
    String buildLine(String command) {


job = job + " " +command;
return job;
    }
    /**
     * Execute la ligne de commande pour le ou les fichiers  imprimer.
     */
    void print() {
ArrayList printLine = new ArrayList();


for (int i = 0; i < filename.size(); i++) {
    job += " "+filename.get(i);
}
final boolean has_files = (filename.size() > 0);
```

```
final boolean debug = (System.getProperty("Debug") != null) ;
try {
    if (debug) {
System.out.println(job);
return ;
    }// debug
    Runtime runtime = Runtime.getRuntime();
    Process process = runtime.exec(job);
    System.out.println("Waiting for \""+job+"\"to finish\n");
    if (! has_files) { // we must do a pipe
OutputStream lp_data = process.getOutputStream();
InputStream data_input =
    new FileInputStream(FileDescriptor.in) ;
int c ;
do {
    c = data_input.read();
    if (c>0) lp_data.write((char) c) ;
} while (c>0);
lp_data.close();
data_input.close();
    } // end of pipe code
    InputStream lp_response = process.getInputStream();

    int c ;
    String result = "" ;
    do {
c = lp_response.read();
if (c>0) { result+=((char) c); }
    } while(c>0);
    GuiFrame2 f2 = new GuiFrame2(result);
    f2.pack();
    f2.setVisible(true);
    int status = process.waitFor();
    if (status<0) {
System.err.println("did not finish correctly");
    } else {
System.out.println("Printer Response: "+result);
    } //
} catch( IndexOutOfBoundsException e ) {
    System.err.println( "Usage:java ExecTest command [args...]" );
    System.exit(-1);
```

```
} catch (InterruptedException e) {
    System.err.println("problem with lp:"+e.getMessage());
    System.exit(-1);
} catch( Exception e ){
    System.err.println( "Error:" + e.toString() );
    System.exit(-1);
}

printLine.clear();


    }

    /**
     * Permets de rajouter un ou des fichiers  imprimer.
     * (S'il y a plusieurs fichiers, les fichiers doivent tre sparer par un
     * point-virgule(;).
     * (Mthode non utilise pour le moment).
     */
    void browseFile(String browsedFile) {

filename.clear();
filename.add(browsedFile);
    }



}
```

A.8 GuiFrame.java

```
/**
 *GuiFrame est une classe qui gnre l'interface graphique du programme
 */
package ch.epfl.lse.impression;

import java.awt.*;
import java.awt.event.*;
import java.text.*;
import java.net.*;

class GuiFrame extends Frame implements ActionListener, WindowListener, ItemListene

    Panel panelWest, panelSouth, panelNorth, paneleEast,  panelCentral;
    Panel panelSouth1, panelSouth2;
    Panel panelGeneral, panelPrintRange, panelMedia;
    Panel panelCentral12, panelCentral13, panelCentral14;
    Panel panelCentral22, panelCentral23, panelCentral24, panelCentral25;
    Panel panelCentral31, panelCentral32, panelCentral33, panelCentral34, panelCent
    Label labelNorth1, labelNorth2, labelWest, labelGeneral, labelPrintRange, label
    Label labelDuplex, labelNumberPages;
    Label labelTo, labelOfN, labelNumberCopies;
    Label labelComment1, labelComment2;
    Label labelMediaSource, labelMediaSize, labelMediaType, labelMediaOutput;
    Label labelRequest;
    List listPrinters;
    Choice choiceDuplex, choiceNumberPages;
    Choice choiceMediaSource, choiceMediaSize, choiceMediaType, choiceMediaOutput;
    Button buttonPrint, buttonCancel;
    Checkbox checkboxPortrait, checkboxLandscape;
    Checkbox checkboxAll, checkboxFrom, checkboxOdd, checkboxEven;
    Checkbox checkboxEconomy, checkboxNoBanner;
    TextField textfieldFrom, textfieldTo, textfieldNumberCopies;
    PrinterList printerList = new PrinterList();
    OptionsGroup optionsGroup=  new OptionsGroup();
    Spooler spooler = new Spooler();
    Job job;
    Options options;
    Collator collator = Collator.getInstance();
    LogoCanvas logoCanvas;
```

```
    PageCount pageCount;
    Font f1 = new Font("Lucida Handwriting",Font.BOLD, 16);
    Font f3 = new Font("Lucida Handwriting",Font.BOLD, 14);
    Font f2 = new Font("TimesRoman",Font.BOLD, 14);
    Font f4 = new Font("TimesRoman", Font.ITALIC,14);
    Color w = Color.white;


    GuiFrame(Job job) {
this.job = job;
options = new Options(job);
Toolkit tk = Toolkit.getDefaultToolkit();
Dimension d = tk.getScreenSize();

pageCount = new PageCount(job);

//Initialisation de la fentre
setLocation(d.width/20,d.height/20);

//Layout manager pour la fentre; BorderLayout
BorderLayout bl = new BorderLayout();
setLayout(bl);

//partie nord
panelNorth = new Panel();
panelNorth.setBackground(w);
FlowLayout flNorth = new FlowLayout(FlowLayout.LEFT);
panelNorth.setLayout(flNorth);
add(panelNorth, "North");
        logoCanvas = new LogoCanvas();
        logoCanvas.setSize(120,34);
labelNorth1 = new Label("-LSE");
        labelNorth2 = new Label("Printer manager");
labelNorth1.setFont(f1);
labelNorth2.setFont(f1);
        panelNorth.add(logoCanvas);
panelNorth.add(labelNorth1);
panelNorth.add(labelNorth2);

//partie ouest
panelWest = new Panel();
```

```
GridLayout glWest = new GridLayout(2,0);
panelWest.setLayout(glWest);
add(panelWest, "West");
labelWest = new Label("Printers:");
labelWest.setFont(f3);
Panel panelWest2 = new Panel();
FlowLayout flWest2 = new FlowLayout();
panelWest2.setLayout(flWest2);
int listPrintersSize = printerList.printerNameList.size();
listPrinters = new List(listPrintersSize+1);
for(int i=0;i<printerList.printerNameList.size();i++)
    listPrinters.add((String)printerList.printerNameList.get(i));
panelWest.add(labelWest);
panelWest2.add(listPrinters);
panelWest.add(panelWest2);
listPrinters.addItemListener(this);

//partie sud
panelSouth = new Panel();
GridLayout glSouth= new GridLayout(2,0);
panelSouth.setLayout(glSouth);
add(panelSouth, "South");

panelSouth1 = new Panel();
panelSouth.add(panelSouth1);
FlowLayout flSouth1 = new FlowLayout();
panelSouth1.setLayout(flSouth1);
buttonPrint = new Button("Print");
buttonPrint.addActionListener(this);
buttonPrint.setEnabled(false);
buttonCancel = new Button("Cancel");
buttonCancel.addActionListener(this);
panelSouth1.add(buttonPrint);
panelSouth1.add(buttonCancel);

panelSouth2 = new Panel();
panelSouth.add(panelSouth2);
FlowLayout flSouth2 = new FlowLayout();
panelSouth2.setLayout(flSouth2);
labelComment1 = new Label("document to print: ");
panelSouth2.add(labelComment1);
```

```
labelComment2 = new Label(job.files);
labelComment2.setFont(f4);
panelSouth2.add(labelComment2);

//partie central
panelCentral = new Panel();
FlowLayout fl= new FlowLayout(FlowLayout.LEFT);
panelCentral.setLayout(fl);
add(panelCentral, "Center");
panelGeneral = new Panel();
panelPrintRange = new Panel();
panelMedia = new Panel();
panelCentral.add(panelGeneral);
panelCentral.add(panelPrintRange);
panelCentral.add(panelMedia);

//Panel: General Options
GridLayout glCentral1 = new GridLayout(4,0);
panelGeneral.setLayout(glCentral1);
labelGeneral = new Label("General Options:");
labelGeneral.setFont(f2);
panelGeneral.add(labelGeneral);

panelCentral12 = new Panel();
panelGeneral.add(panelCentral12);
FlowLayout fl12= new FlowLayout(FlowLayout.LEFT);
panelCentral12.setLayout(fl12);
checkboxPortrait = new Checkbox("Portrait", true);
checkboxLandscape = new Checkbox("Landscape", false);
panelCentral12.add(checkboxPortrait);
checkboxPortrait.setEnabled(false);
        checkboxPortrait.addItemListener(this);
panelCentral12.add(checkboxLandscape);
checkboxLandscape.setEnabled(false);
        checkboxLandscape.addItemListener(this);

panelCentral13 = new Panel();
panelGeneral.add(panelCentral13);
FlowLayout fl13= new FlowLayout(FlowLayout.LEFT);
panelCentral13.setLayout(fl13);
labelDuplex = new Label("Duplex:");
```

```
panelCentral13.add(labelDuplex);
choiceDuplex = new Choice();
panelCentral13.add(choiceDuplex);
choiceDuplex.addItemListener(this);
choiceDuplex.setEnabled(false);

panelCentral14 = new Panel();
panelGeneral.add(panelCentral14);
FlowLayout fl14= new FlowLayout(FlowLayout.LEFT);
panelCentral14.setLayout(fl14);
labelNumberPages = new Label("Number of pages per sheet:");
panelCentral14.add(labelNumberPages);
choiceNumberPages = new Choice();
panelCentral14.add(choiceNumberPages);
choiceNumberPages.addItemListener(this);
choiceNumberPages.setEnabled(false);

//Panel: Print Range
GridLayout glCentral2 = new GridLayout(5,0);
panelPrintRange.setLayout(glCentral2);
labelPrintRange = new Label("Print Range:");
panelPrintRange.add(labelPrintRange);
labelPrintRange.setFont(f2);

panelCentral22 = new Panel();
FlowLayout fl22 = new FlowLayout(FlowLayout.LEFT);
panelCentral22.setLayout(fl22);
checkboxAll = new Checkbox("All", true);
panelCentral22.add(checkboxAll);
checkboxAll.setEnabled(false);
        checkboxAll.addItemListener(this);
panelPrintRange.add(panelCentral22);

panelCentral23 = new Panel();
panelPrintRange.add(panelCentral23);
FlowLayout fl23 = new FlowLayout(FlowLayout.LEFT);
panelCentral23.setLayout(fl23);
checkboxFrom = new Checkbox("From:", false);
textfieldFrom = new TextField("   ");
textfieldFrom.setEnabled(false);
labelTo = new Label("To",Label.CENTER);
```

```
textfieldTo = new TextField("   ");
textfieldTo.setEnabled(false);
String s = pageCount.getnbPages();
if (collator.equals(s,""))
    labelOfN = new Label(s,Label.CENTER);
else
    labelOfN = new Label("of "+ s,Label.CENTER);
panelCentral23.add(checkboxFrom);
checkboxFrom.setEnabled(false);
checkboxFrom.addItemListener(this);
panelCentral23.add(textfieldFrom);
textfieldFrom.addActionListener(this);
panelCentral23.add(labelTo);
panelCentral23.add(textfieldTo);
textfieldTo.addActionListener(this);
panelCentral23.add(labelOfN);

panelCentral24 = new Panel();
panelPrintRange.add(panelCentral24);
FlowLayout fl24 = new FlowLayout(FlowLayout.LEFT);
panelCentral24.setLayout(fl24);
checkboxOdd = new Checkbox("Odd", true);
checkboxEven = new Checkbox("Even", true);
panelCentral24.add(checkboxOdd);
checkboxOdd.addItemListener(this);
checkboxOdd.setEnabled(false);
panelCentral24.add(checkboxEven);
checkboxEven.addItemListener(this);
checkboxEven.setEnabled(false);

panelCentral25 = new Panel();
panelPrintRange.add(panelCentral25);
FlowLayout fl25 = new FlowLayout(FlowLayout.LEFT);
panelCentral25.setLayout(fl25);
labelNumberCopies = new Label("number of copies");
textfieldNumberCopies = new TextField("   ");
textfieldNumberCopies.setEnabled(false);
panelCentral25.add(labelNumberCopies);
panelCentral25.add(textfieldNumberCopies);
textfieldNumberCopies.addActionListener(this);
```

```
//Panel: Media Settings
GridLayout glCentral3 = new GridLayout(6,0);
panelMedia.setLayout(glCentral3);
labelMedia = new Label("Media Settings:");
panelMedia.add(labelMedia);
labelMedia.setFont(f2);

panelCentral31 = new Panel();
FlowLayout fl31 = new FlowLayout(FlowLayout.LEFT);
panelCentral31.setLayout(fl31);
checkboxEconomy = new Checkbox("Economy");
panelCentral31.add(checkboxEconomy);
checkboxEconomy.addItemListener(this);
checkboxEconomy.setEnabled(false);
checkboxNoBanner = new Checkbox("No Banner");
panelCentral31.add(checkboxNoBanner);
checkboxNoBanner.addItemListener(this);
checkboxNoBanner.setEnabled(false);
panelMedia.add(panelCentral31);


panelCentral32 = new Panel();
panelMedia.add(panelCentral32);
FlowLayout fl32 = new FlowLayout(FlowLayout.LEFT);
panelCentral32.setLayout(fl32);
labelMediaSource = new Label("Media Source:");
choiceMediaSource = new Choice();
panelCentral32.add(labelMediaSource);
panelCentral32.add(choiceMediaSource);
choiceMediaSource.addItemListener(this);
choiceMediaSource.setEnabled(false);

panelCentral33 = new Panel();
panelMedia.add(panelCentral33);
FlowLayout fl33 = new FlowLayout(FlowLayout.LEFT);
panelCentral33.setLayout(fl33);
labelMediaSize = new Label("Media Size:");
choiceMediaSize = new Choice();
panelCentral33.add(labelMediaSize);
panelCentral33.add(choiceMediaSize);
choiceMediaSize.addItemListener(this);
```

```
choiceMediaSize.setEnabled(false);

panelCentral34 = new Panel();
panelMedia.add(panelCentral34);
FlowLayout fl34 = new FlowLayout(FlowLayout.LEFT);
panelCentral34.setLayout(fl34);
labelMediaType = new Label("Media Type:");
choiceMediaType = new Choice();
panelCentral34.add(labelMediaType);
panelCentral34.add(choiceMediaType);
choiceMediaType.addItemListener(this);
choiceMediaType.setEnabled(false);

panelCentral35 = new Panel();
panelMedia.add(panelCentral35);
FlowLayout fl35 = new FlowLayout(FlowLayout.LEFT);
panelCentral35.setLayout(fl35);
labelMediaOutput = new Label("Media Output:");
choiceMediaOutput = new Choice();
panelCentral35.add(labelMediaOutput);
panelCentral35.add(choiceMediaOutput);
choiceMediaOutput.addItemListener(this);
choiceMediaOutput.setEnabled(false);

this.addWindowListener(this);

    }


    public void windowClosing(WindowEvent e) {
System.exit(0);;
    }
    public void windowOpened(WindowEvent e){}
    public void windowClosed(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
    public void windowActivated(WindowEvent e){}
    public void windowDeactivated(WindowEvent e){}

    /**
     * mthode qui ragit au action des diffrents boutons de l'interface graphique
```

```
     * @param e l'vnement correspondant  une action sur un bouton de l'interface gr
     */
    public void actionPerformed(ActionEvent e) {
Object o = e.getSource();
if(o instanceof Button) {
    Button b = (Button)o;
    if(b == buttonCancel)
System.exit(0);
    if(b == buttonPrint) {
if (checkboxLandscape.getState())
    job.orientation = spooler.checkboxLandscape;
else
    job.orientation = spooler.checkboxPortrait;
job.duplex = choiceDuplex.getSelectedItem();
job.numberPages = choiceNumberPages.getSelectedItem();
if (checkboxFrom.getState()) {
    job.range = true;
    job.printFrom = textfieldFrom.getText();
    job.printTo = textfieldTo.getText();
}
else
    job.range = false;
job.odd = checkboxOdd.getState();
job.even = checkboxEven.getState();
job.economy = checkboxEconomy.getState();
job.noBanner = checkboxNoBanner.getState();
job.numberCopies = textfieldNumberCopies.getText();
job.mediaSource = choiceMediaSource.getSelectedItem();
job.mediaSize = choiceMediaSize.getSelectedItem();
job.mediaType = choiceMediaType.getSelectedItem();
job.mediaOutput = choiceMediaOutput.getSelectedItem();
job.printerName = listPrinters.getSelectedItem();
options.sendToJob();
this.hide();
    }
}
    }
    /**
     * mthode qui ragit au action des diffrentes listes de l'interface graphique
     * @param e l'vnement correspondant  un changement de selection d'une liste de
     */
```

```
    public void itemStateChanged (ItemEvent e) {
Object o = e.getSource();
if(o instanceof List) {
    List l = (List)o;
    if (l == listPrinters) {
String labelChoice = l.getSelectedItem();
printerList.createPrinter(labelChoice);
choiceDuplex.removeAll();
choiceNumberPages.removeAll();
choiceMediaSource.removeAll();
choiceMediaSize.removeAll();
choiceMediaType.removeAll();
choiceMediaOutput.removeAll();
optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.choiceDuplex);
for(int i=0;i<optionsGroup.optionsList.size();i++)
    choiceDuplex.add((String)optionsGroup.optionsList.get(i));
choiceDuplex.setEnabled(true);

optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.choiceNumberPages);
for(int i=0;i<optionsGroup.optionsList.size();i++)
    choiceNumberPages.add((String)optionsGroup.optionsList.get(i));
choiceNumberPages.setEnabled(true);

optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.choiceMediaSource);
for(int i=0;i<optionsGroup.optionsList.size();i++)
    choiceMediaSource.add((String)optionsGroup.optionsList.get(i));
choiceMediaSource.setEnabled(true);

optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.choiceMediaSize);
for(int i=0;i<optionsGroup.optionsList.size();i++)
    choiceMediaSize.add((String)optionsGroup.optionsList.get(i));
choiceMediaSize.setEnabled(true);

optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.choiceMediaType);
for(int i=0;i<optionsGroup.optionsList.size();i++)
    choiceMediaType.add((String)optionsGroup.optionsList.get(i));
```

```
choiceMediaType.setEnabled(true);

optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.choiceMediaOutput);
for(int i=0;i<optionsGroup.optionsList.size();i++)
    choiceMediaOutput.add((String)optionsGroup.optionsList.get(i));
choiceMediaOutput.setEnabled(true);

                checkboxPortrait.setEnabled(true);
                checkboxPortrait.setState(true);
                checkboxAll.setEnabled(true);
                checkboxAll.setState(true);
                buttonPrint.setEnabled(true);
                textfieldNumberCopies.setEnabled(true);
textfieldNumberCopies.setText("1");
                checkboxLandscape.setEnabled(false);
                checkboxLandscape.setState(false);
optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.orientation);
for(int i=0;i<optionsGroup.optionsList.size();i++) {
    if( collator.equals((String)optionsGroup.optionsList.get(i),spooler.checkboxLan
checkboxLandscape.setEnabled(true);
}
                optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.pageRange);
                checkboxFrom.setEnabled(false);
                checkboxFrom.setState(false);
                textfieldFrom.setEnabled(false);
                textfieldFrom.setText("");
                textfieldTo.setEnabled(false);
                textfieldTo.setText("");
checkboxOdd.setEnabled(false);
checkboxEven.setEnabled(false);
                checkboxOdd.setState(true);
                checkboxEven.setState(true);
checkboxEconomy.setEnabled(false);
checkboxNoBanner.setEnabled(false);
checkboxNoBanner.setState(false);
checkboxEconomy.setState(false);
                for(int i=0;i<optionsGroup.optionsList.size();i++) {
    if(collator.equals((String)optionsGroup.optionsList.get(i),spooler.checkboxFrom
```

```
                        checkboxFrom.setEnabled(true);
                }
                for(int i=0;i<optionsGroup.optionsList.size();i++) {
    if(collator.equals((String)optionsGroup.optionsList.get(i),spooler.checkboxOdd)
checkboxOdd.setEnabled(true);
}
                for(int i=0;i<optionsGroup.optionsList.size();i++) {
    if(collator.equals((String)optionsGroup.optionsList.get(i),spooler.checkboxEven
checkboxEven.setEnabled(true);
}
optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.checkboxEconomy);
for(int i=0;i<optionsGroup.optionsList.size();i++) {
    if(collator.equals((String)optionsGroup.optionsList.get(i),spooler.checkboxEcon
                        checkboxEconomy.setEnabled(true);
                }
optionsGroup.clearListOptions();
optionsGroup.listOptions(spooler.checkboxNoBanner);
for(int i=0;i<optionsGroup.optionsList.size();i++) {
    if(collator.equals((String)optionsGroup.optionsList.get(i),spooler.checkboxNoBa
                        checkboxNoBanner.setEnabled(true);
                }
pack();

    }
}
if (o instanceof Checkbox) {
    Checkbox cb = (Checkbox)o;
    if (cb == checkboxPortrait) {
boolean statePortrait = cb.getState();
                if (statePortrait)
                    checkboxLandscape.setState(false);
                else
                    checkboxPortrait.setState(true);
    }
    if (cb == checkboxLandscape) {
boolean stateLandscape = cb.getState();
if (stateLandscape)
    job.orientation = spooler.checkboxLandscape;
if (stateLandscape)
    checkboxPortrait.setState(false);
```

```
                else
                    checkboxLandscape.setState(true);
    }
              if (cb == checkboxAll) {
                 boolean stateAll = cb.getState();
                 if (stateAll)
                     checkboxFrom.setState(false);
                 else
                     checkboxAll.setState(true);
                 textfieldFrom.setEnabled(false);
                 textfieldTo.setEnabled(false);
              }
              if (cb == checkboxFrom) {
              boolean stateFrom = cb.getState();
              if (stateFrom)
                   checkboxAll.setState(false);
              else
                   checkboxFrom.setState(true);
               textfieldFrom.setEnabled(true);
               textfieldTo.setEnabled(true);
              }

}
    }

}


//-----------------------------------------------------------------

/**
 *GuiCanvas est une classe qui gnre le logo EPFL prsents dans l'interface graphique
 */
class LogoCanvas extends Canvas {

    Image logo;

    LogoCanvas() {
Toolkit tk = Toolkit.getDefaultToolkit();
Class pl = this.getClass();
URL logoURL = pl.getResource("logo.gif");
```

```
logo = tk.getImage(logoURL);
    }


    public void paint (Graphics g) {
        g.drawImage(logo,0,0,this);
    }
}


//-----------------------------------------------------------------
/**
 *GuiFrame2 est une classe qui gnre une fentre affichant le flux sortant de l'impri
 */
class GuiFrame2 extends Frame implements WindowListener, ActionListener {

    Label l;
    Button b;
    Panel p1, p2;

    GuiFrame2(String s) {
Toolkit tk = Toolkit.getDefaultToolkit();
Dimension d = tk.getScreenSize();
setLocation(d.width/2,d.height/2);
GridLayout gl = new GridLayout(2,0);
setLayout(gl);


p1 = new Panel();
FlowLayout fl1 = new FlowLayout();
p1.setLayout(fl1);
l = new Label(s);
p1.add(l);

p2 = new Panel();
FlowLayout fl2 = new FlowLayout();
p2.setLayout(fl2);
b = new Button("  OK  ");
b.addActionListener(this);
p2.add(b);
```

```
add(p1);
add(p2);

this.addWindowListener(this);
    }

    public void windowClosing(WindowEvent e) {
this.hide();
    }
    public void windowOpened(WindowEvent e){}
    public void windowClosed(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
    public void windowActivated(WindowEvent e){}
    public void windowDeactivated(WindowEvent e){}

    public void actionPerformed(ActionEvent e) {
Object o = e.getSource();
if(o instanceof Button) {
    System.exit(0);
}
    }
}
```

A.9 PageCount.java

```
package ch.epfl.lse.impression;

import java.lang.*;
import java.io.*;
import java.util.*;
import java.net.*;

/** PageCount utilise quelques commandes UNIX pour retrouver des informations utile
 * aux documents  imprimer ou au fonctionnement de lp.
 */
class PageCount {
    Spooler spooler = new Spooler();
    Job job;
    /** commande pour retrouver le nombre de pages dans un document PS */
    String grep = "grep  %%Pages: ";
    /** variable contenant le nombre de pages du document  imprimer */
    String nbPages = "";

    PageCount(Job job) {
this.job = job;
    }
    /**
     * Cherche le nombre de pages du document envoy  la ligne de commande
     * avec la commande UNIX "grep".
     * @return le nombre de pages du document
     */
    String getnbPages() {


for (int i = 0; i < job.filename.size(); i++) {
    String file  = (String)job.filename.get(i);

    String command = grep+ " " + file;

    try {
Runtime runtime = Runtime.getRuntime();
Process process = runtime.exec(command);
InputStream grep_response = process.getInputStream();
```

```
int c;
    String result = "";
do{
    c = grep_response.read();
    if(c>0) {result +=((char) c);}
}while(c>0);

StringTokenizer st = new StringTokenizer(result, "%%Pages: ");

while (st.hasMoreTokens()) {
    nbPages = (String)st.nextToken();
}

    }catch ( IOException e) {System.out.println("problem with grep");}


}
return nbPages;
    }

    /**
     * Cherche le chemin pour la commande lp par la commande UNIX "which".
     * Ecrit le chemin trouv dans le fichier lpPath.dat
     * @return chemin de lp
     */

    String getlpPath() {
            String result = "";
    String command = "which lp";

    try {
Runtime runtime = Runtime.getRuntime();
Process process = runtime.exec(command);
InputStream which_response = process.getInputStream();

int c;

do{
    c = which_response.read();
    if(c>0) {result +=((char) c);}
}while(c>0);
```

```
    }catch ( IOException e) {System.out.println("problem with getting lp Path");}

            try {
FileWriter fw = new FileWriter("lpPath.dat");
fw.write(result);
fw.close();
    }
catch(IOException e) {
    System.out.println("Error while writing in lpPath.dat");
}


return result;
    }

    /** Charge le Path de lp depuis le fichier lpPath.dat */

    String loadlpPath() {

    String lpPath = "";
    try {
        Class pl = this.getClass();
        InputStream file = pl.getResourceAsStream(spooler.lpPath);

        int c;
do {
        c = file.read();
        if (c>0)
        lpPath +=((char) c);
        }while (c>0);
        file.close();
}
catch(IOException e) {
    System.out.println("Unable to read from file " +spooler.lpPath);
}
        return lpPath;
    }

}
```

B Fichiers de Configuration

B.1 hp2100

```
#Options of hp2100 printers
#
#Properties for Usual Options
Select_printer=PrintTo
Number_of_copies=MultipleCopies
#
#Properties for general Options
Orientation=Portrait,Landscape
Duplex=default,1-Sided,2-Sided_long_edge,2-Sided_short_edge,2-Sided_manual
#
#Properties for media types
Media_size=default,A4,A3,A5,A6,B5-ISO,Legal,Letter,Executive
Media_type=default,Plain,Preprinted,Letterhead,Transparency,Prepunched,Labels,Bond,
Media_source=default,tray_1,tray_2,tray_3,manual
Media_output=default,bin1,bin2
Economy=Economy
No_banner=No_banner
#
#Properties for document options
Page_ranges=All,From_to
Number_of_pages_per_sheet=default,One_per_sheet,Two_per_sheet,Four_per_sheet,Eight_
```

```
#Command List for printers hp4100
#
default=
choosePrinter=-d
MultipleCopies=-n
#ORIENTATION
Portrait= -o portrait
Landscape=-o landscape
#SIMPLEX AND DUPLEX
1-Sided=-o simplex
2-Sided_long_edge=-o duplex
2-Sided_short_edge=-o hduplex
2-Sided_manual=-o mduplex
#MEDIA SIZE
A4=-o A4
A3=-o A3
A5=-o A5
A6=-o A6
B5-ISO=-o B5-JIS
Legal=-o legal
Letter=-o letter
Executive=-o exec
#MEDIA TYPE
Plain=-o mtypePlain
Preprinted=-o mtypePreprinted
Letterhead=-o mtypeLetterhead
Transparency=-o mtypeTransparency
Prepunched=-o mtypePrepunched
Labels=-o mtypeLabels
Bond=-o mtypeBond
Recycled=-o mtypeRecycled
Color=-o mtypeColor
CardStock=-o mtypeCardStock
Rough=-o mtypeRough
Vellum=-o mtypeVellum
#MEDIA SOURCE
manual= -o manual
tray_1=-o tray1
tray_2=-o tray2
tray_3=-o tray3
#MEDIA OUTPUT
```

```
bin1=-o bin1
bin2=-o bin2
#NUP
Two_per_sheet=-o 2up
Four_per_sheet=-o 4up
Eight_per_sheet=-o 8up
#
#
Economy=-o econoon
Odd= -o odd
Even=-o even
From_to=-P
No_banner=-o nb
```

B.2 hp4100

```
#Options of hp2100 printers
#
#Properties for Usual Options
Select_printer=PrintTo
Number_of_copies=MultipleCopies
#
#Properties for general Options
Orientation=Portrait,Landscape
Duplex=default,1-Sided,2-Sided_long_edge,2-Sided_short_edge,2-Sided_manual
#
#Properties for media types
Media_size=default,A4,A3,A5,A6,B5-ISO,Legal,Letter,Executive
Media_type=default,Plain,Preprinted,Letterhead,Transparency,Prepunched,Labels,Bond,
Media_source=default,tray_1,tray_2,tray_3,manual
Media_output=default,bin1,bin2
Economy=Economy
No_banner=No_banner
#
#Properties for document options
Page_ranges=All,From_to
Number_of_pages_per_sheet=default,One_per_sheet,Two_per_sheet,Four_per_sheet,Eight_
```

```
#Command List for printers hp4100
#
default=
choosePrinter=-d
MultipleCopies=-n
#ORIENTATION
Portrait= -o portrait
Landscape=-o landscape
#SIMPLEX AND DUPLEX
1-Sided=-o simplex
2-Sided_long_edge=-o duplex
2-Sided_short_edge=-o hduplex
2-Sided_manual=-o mduplex
#MEDIA SIZE
A4=-o A4
A3=-o A3
A5=-o A5
A6=-o A6
B5-ISO=-o B5-JIS
Legal=-o legal
Letter=-o letter
Executive=-o exec
#MEDIA TYPE
Plain=-o mtypePlain
Preprinted=-o mtypePreprinted
Letterhead=-o mtypeLetterhead
Transparency=-o mtypeTransparency
Prepunched=-o mtypePrepunched
Labels=-o mtypeLabels
Bond=-o mtypeBond
Recycled=-o mtypeRecycled
Color=-o mtypeColor
CardStock=-o mtypeCardStock
Rough=-o mtypeRough
Vellum=-o mtypeVellum
#MEDIA SOURCE
manual= -o manual
tray_1=-o tray1
tray_2=-o tray2
tray_3=-o tray3
#MEDIA OUTPUT
```

```
bin1=-o bin1
bin2=-o bin2
#NUP
Two_per_sheet=-o 2up
Four_per_sheet=-o 4up
Eight_per_sheet=-o 8up
#
#
Economy=-o econoon
Odd= -o odd
Even=-o even
From_to=-P
No_banner=-o nb
```

B.3 lps17

```
#Options of type lps17 printers
#
#Properties for Usual Options
Select_printer=choosePrinter
Number_of_copies=sheetcount
No_banner=No_banner
Economy=
#
#Properties for general Option
Orientation=Portrait,Landscape
Duplex=default,1-Sided,2-Sided
#
#Properties for media types
Media_size=default,A4,A5,A3,B5,B4,letter,ledger,legal,Executive,US10_envelope,DL_en
Media_type=default,Letterhead_paper,Drilled_paper,Transparencies,Gummed_labels
Media_source=default,Upper_input_tray,Bottom_input_tray,Large_capacity_input_tray,M
Media_output=default,Side_tray(face_up),Lower_tray(face_down),Mail_box,Side_tray,To
#
#Properties for document options
Page_ranges=All,From_to
Number_of_pages_per_sheet=default,One_per_sheet,Two_per_sheet,Four_per_sheet
```

```
#Command List for printers of type lps17
#
default=
#INTRAY
Upper_input_tray=-o intray\=top
Bottom_input_tray=-o intray\=bottom
Large_capacity_input_tray=-o intray\=lcit
Manual_feed_slot=-o intray\=manual
Envelope_tray=-o intray\=envelope
#
#NUP
One_per_sheet=-o nup=1
Two_per_sheet=-o nup=2
Four_per_sheet=-o nup=4
#
#OUTTRAY
Side_tray(face_up)=-o outtray\=face-up
Lower_tray(face_down)=-o outtray\=lower
Mail_box=-o outtray\=mb
Side_tray=-o outtray\=side
Top_tray=-o outtray\=top
Upper_tray=-o outtray\=upper
#
#SHEETCOUNT
sheetcount=-o sheetcount\=
#
#MEDIA SIZE
ISO_A4=-o sheetsize\=a4
ISO_A5=-o sheetsize\=a5
ISO_A3=-o sheetsize\=a3
ISO_B5=-o sheetsize\=b5
ISO_B4=-o sheetsize\=b4
US_letter=-o sheetsize\=letter
US_ledger=-o sheetsize\=ledger
US_legal=-o sheetsize\=legal
Executive=-o sheetsize\=executive
US10_envelope=-o sheetsize\=business_envelope
ISO_DL_envelope=-o sheetsize\=dl
7_envelope=-o sheetsize\=7_envelope
#
#SIMPLEX AND DUPLEX
```

```
1-Sided=-o simplex
2-Sided=-o duplex
#
#ORIENTATION
Portrait=-o orient\=port
Landscape=-o orient\=land
#
#SPECIAL MEDIA
Letterhead_paper=
Drilled_paper=
Transparencies=-o face-up
Gummed_labels=-o face-up
#
#
choosePrinter=-d
Odd=-o page-set\=odd
Even=-o page-set\=even
From_to=-P
No_banner=-o nobanner
```

B.4 lp Path

```
/usr/bin/lp
```